UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 10/687,941 | 10/17/2003 | Abdul Kayam | 24043-08537 | 9901 |

758 7590 09/14/2007

FENWICK & WEST LLP
SILICON VALLEY CENTER
801 CALIFORNIA STREET
MOUNTAIN VIEW, CA 94041

| EXAMINER | |
|---|---|
| WANG, BEN C | |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2192 | |

| MAIL DATE | DELIVERY MODE |
|---|---|
| 09/14/2007 | PAPER |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

MN

| Office Action Summary | Application No. | Applicant(s) | |
|---|---|---|---|
| | 10/687,941 | KAYAM ET AL. | |
| | Examiner | Art Unit | |
| | Ben C. Wang | 2192 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on *09 July 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) *1-14,17,20-50 and 62-64* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-14,17,20-50 and 62-64* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

## DETAILED ACTION

1. Applicant's amendment dated July 9, 2007, responding to the Office action mailed February 9, 2007 provided in the rejection of claims 1-61, wherein claims 1,11, 36-39, 41-443, and 45-48 are amended, claims 15-16, 18-19, and 51-61 are canceled, and claims 62-64 are new.

Claims 1-14, 17, 20-50, and 62-64 remain pending in the application and which have been fully considered by the examiner.

Applicant's arguments with respect to claims rejection have been fully considered but are moot in view of the new grounds of rejection – see *Thilmany et al., Lee et al.*, and *Moore et al.*, arts made of record, as applied hereto.

Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

### *Claim Objections*

2. Claims 2-14, 17, 20-35, 37-48, and 62 are objected to because the following informalities:

- "A method according to claim", recited in claims 2-14, 17, and 20-35, should be corrected as "The method according to claim".
- "A computer system according to claim", recited in claims 37-48, should be corrected as "The computer system according to claim".
- "A method according to claim 1", recited in claim 62, should be corrected as "The method according to claim 1".

Appropriate correction is required.

### *Claim Rejections – 35 USC § 102(b)*

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102(b) that form the basis for the rejections under this section made in this office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

3. Claims 1-9, 20-26, 28-50, and 62-64are rejected under 35 U.S.C. 102(b) as being anticipated by Thilmany et al. (*BizTalk®: Implement Design Patterns for Business Rules with Orchestration Designer, Oct. 2001, MSDN® Magazine*) (hereinafter 'Thilmany' - art made of record)

4. **As to claim 1** (Currently Amended), Thilmany discloses a method of creating an application for executing on at least one machine having a memory, the method comprising:

- creating a definition of at least one node and a specification which are held in at least one machine readable data file (e.g., Sec. of BizTalk® and the Orchestration Designer®, 3rd Par. – the Biztalk Orchestration Designer® is a Microsoft Visio® 2000-based design tool that, when compiled, can run as an XLANG® schedule; XLANG® itself is described through XML);
- the specification defining how the at least one node can interact with other nodes (e.g., Sec. of the chain of responsibility pattern, 1st Par. – the intent of the Chain of Responsibility pattern is to provide a loosely coupled sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes the request along the chain until the request is handled; Sec. of Using the BizTalk Orchestration Designer®, 2nd Par. – ports are named locations where messages are sent and received; ports can be defined initially as unbound or bound), resources useable by the at least one node, at least one set of predetermined rules to be used by the node for processing data (e.g., Sec. of Using the BizTalk Orchestration Designer®, 1st Par. – there are three key object categories in the business process page of the Orchestration Designer®; flowchart shapes, ports, and implementation shapes; conceptually, the way this works is that you, or your friendly neighborhood business analyst, use the flowchart shapes to create the

process flow; the flowchart shapes support basic constructs such as if ... then ... else decisions, looping while a condition is true, branching execution, rejoining the branches together, and defining transactional boundaries) and messages that can be passed between nodes such that the node is capable of receiving messages in any format which can trigger a rule if predetermined data is present (e.g., Sec. of BizTalk® and the Orchestration Designer®, 2nd Par. – communicating to a BizTalk® server implementation simply means using XML as the message format; as you may already know, BizTalk® is completely structured around XML and uses a SOAP 1.1 XML message format; the BizTalk® friendly XML message is nothing more than a SOAP 1.1 XML message with additional biz-tags to comply with the BizTalk® Framework specification);

- providing a run time environment which processes the at least one machine readable data file in order to implement the at least one node within the memory of the machine such that the node is capable of receiving data and processing received data according to the set of predetermined rules and outputting the processed data (e.g., Sec. of the chain of responsibility pattern, 1st Par. – the intent of the Chain of Responsibility pattern is to provide a loosely coupled sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes the request along the chain until the request is handled; Sec. of Using the BizTalk Orchestration Designer®, 2nd Par. – ports are named locations where messages are sent and received; ports can be defined initially as unbound or

bound; Sec. of BizTalk and the Orchestration Designer®, 3rd par. – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of a message queue, COM component, BizTalk® Channel, e-mail message, or HTTP-based service; orchestration designer provides a way to create loosely coupled business processes that may optionally be long-running a nature);

- wherein the processing of the machine readable data file, in the run time environment, interconnects each node according to the definition provided in the specification such that data input to the application is processed by the at least one node and, if further processing is required, forwarded to other nodes for that further processing (e.g., Sec. of the chain of responsibility pattern, 1st Par. – the intent of the Chain of Responsibility pattern is to provide a loosely coupled sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes the request along the chain until the request is handled; Sec. of Using the BizTalk Orchestration Designer®, 2nd Par. – ports are named locations where messages are sent and received; ports can be defined initially as unbound or bound).

5. **As to claim 2** (incorporating the rejection in claim 1) (Original), Thilmany discloses a (the) method in which a plurality of nodes are created (e.g., Sec. of Using the BizTalk Orchestration Designer®, 2nd Par. – Ports are named locations where messages are sent and received; ports can be defined initially as unbound or bound; 3rd

Par. – Binding a port provides the ability to define precisely what will travel through the port in each direction, since the target implementation is know when the binding is created).

6. **As to claim 3** (incorporating the rejection in claim 1) (Original), Thilmany discloses a (the) method which further comprises providing a library of nodes containing at least one node and selecting at least one of the nodes from the library of nodes (e.g., P. 1, 6th Par. – the sample application will show how applying patterns to Biztalk® can significantly ease the development and design for each sub-department; it will provide the ability to answer and route online product support questions effectively and efficiently; Sec. of Design Patterns versus Architectural Patterns, 1st Par. – design patterns help make the design process faster; this allows solution providers to take the time to concentrate on the business implementation; more importantly, patterns help formalize the design to make it reusable; reusability not only applies to the components themselves, but also the stages the design must go through to morph itself into your final solution; Sec. of Biztalk® and the Orchestration Designer®, 3rd Par. – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of a message queue, COM component, BizTalk® Channel, file, e-mail message, or HTTP-based service; Orchestration Designer provides a way to create loosely coupled business processes that may optionally be long-running in nature).

7. **As to claim 4** (incorporating the rejection in claim 1) (Original), Thilmany discloses a (the) method which further comprises arranging the or each node to comprise a plurality of layers, each layer being arranged to perform a predetermined function (e.g., Sec. of Biztalk® and the Orchestration Designer®, 3rd Par. – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of a message queue, COM component, BizTalk® Channel, file, e-mail message, or HTTP-based service; Orchestration Designer provides a way to create loosely coupled business processes that may optionally be long-running in nature).

8. **As to claim 5** (incorporating the rejection in claim 4) (Original), Thilmany discloses a (the) method which further comprises arranging the layers of the nodes to be interchangeable and wherein altering at least one of the layers can change the overall functionality of a node (e.g., Sec. of Biztalk® and the Orchestration Designer®, 3rd Par. – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of a message queue, COM component, BizTalk® Channel, file, e-mail message, or HTTP-based service; Orchestration Designer® provides a way to create loosely coupled business processes that may optionally be long-running in nature).

9. **As to claim 6** (incorporating the rejection in claim 4) (Original), Thilmany discloses a (the) method which further comprises providing a library of layers containing

at least one layer and selecting at least one layer from the library of layers (e.g., P. 1, 6th Par. – the sample application will show how applying patterns to Biztalk® can significantly ease the development and design for each sub-department; it will provide the ability to answer and route online product support questions effectively and efficiently; Sec. of Design Patterns versus Architectural Patterns, 1st Par. – design patterns help make the design process faster; this allows solution providers to take the time to concentrate on the business implementation; more importantly, patterns help formalize the design to make it reusable; reusability not only applies to the components themselves, but also the stages the design must go through to morph itself into your final solution; Sec. of Biztalk® and the Orchestration Designer®, 3rd Par. – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of a message queue, COM component, BizTalk® Channel, file, e-mail message, or HTTP-based service; Orchestration Designer provides a way to create loosely coupled business processes that may optionally be long-running in nature).

10. **As to claim 7** (incorporating the rejection in claim 4) (Original), Thilmany discloses a (the) method which comprises arranging at least one of the layers of a node to act as a transport layer arranged to receive and send data to and from the node (e.g., Sec. of Using the Biztalk Orchestration Designer®, 2nd Par. – ports are named locations where messages are sent and received; Sec. of Biztalk and the Orchestration Designer, 2nd Par. – Communicating to a Biztalk® server implementation simply means using XML

as the message format; BizTalk is completely structured around XML and uses a SOAP 1.1 XML message format; the Biztalk® friendly XML message is nothing more than a SOAP 1.1 XML message with additional biz-tags to comply with the BizTalk® Framework specification).

11. **As to claim 8** (incorporating the rejection in claim 4) (Original), Thilmany discloses a (the) method which comprises arranging at least one of the layers of a node to act as a message transceiver arranged to send and receive messages to other nodes to which that node is connected (e.g., Sec. of Using the Biztalk Orchestration Designer®, 2nd Par. – ports are named locations where messages are sent and received; Sec. of Biztalk® and the Orchestration Designer®, 2nd Par. – Communicating to a Biztalk® server implementation simply means using XML as the message format; BizTalk® is completely structured around XML and uses a SOAP 1.1 XML message format; the Biztalk® friendly XML message is nothing more than a SOAP 1.1 XML message with additional biz-tags to comply with the BizTalk® Framework specification).

12. **As to claim 9** (incorporating the rejection in claim 8) (Original), Thilmany discloses a (the) method in which the at least one node has an identity and in which the application is arranged to be run and, at runtime, as nodes are connected together, the method further comprising arranging the message transceiver layer of a node to discover the identity of nodes to which it is connected at runtime (e.g., Sec. of the chain of responsibility pattern, 1st Par. – the intent of the Chain of Responsibility pattern is to

provide a loosely coupled sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes the request along the chain until the request is handled; Sec. of Using the BizTalk Orchestration Designer®, 2nd Par. – ports are named locations where messages are sent and received; ports can be defined initially as unbound or bound).

13. **As to claim 20** (incorporating the rejection in claim 1) (Original), Thilmany discloses a (the) method which comprises providing a pattern, arranging the at least one node within the pattern and defining how nodes therein interact with one another (e.g., SUMMARY – this article describes several traditional design patterns including the Observer pattern and the Dispatcher pattern, elaborates on their structures, what they're used for, and how they can help you build a Biztalk®-based solution; Sec. of The Chain of Responsibility Pattern, 1st Par. – the intent of the Chain of Responsibility pattern is to provide a loosely coupled sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes the request along the chain until the request is handled).

14. **As to claim 21** (incorporating the rejection in claim 20) (Original), Thilmany discloses a (the) method which comprises providing a library of patterns containing at least one pattern that can be used in creating an application (e.g., SUMMARY – this article describes several traditional design patterns including the Observer pattern and the Dispatcher pattern, elaborates on their structures, what they're used for, and how

they can help you build a Biztalk-based solution; Sec. of The Chain of Responsibility Pattern, 1st Par. – the intent of the Chain of Responsibility pattern is to provide a loosely coupled sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes the request along the chain until the request is handled; Sec. of Design Patterns versus Architectural Patterns, 1st Par. – Reusability not only applies to the components themselves, but also to the stages the design must go through to morph itself into your final solution; the ability to apply a patterned repeatable solution is worth the little time spent learning formal patterns, or to even formalize you own).

15. **As to claim 22** (incorporating the rejection in claim 1) (Original), Thilmany discloses a (the) method in which the specification is arranged to determine at least one of the following: which nodes are to be used; which nodes interact with one another; which patterns are to be used; which assets are to be used (e.g., SUMMARY – this article describes several traditional design patterns including the Observer pattern and the Dispatcher pattern, elaborates on their structures, what they're used for, and how they can help you build a Biztalk-based solution; Sec. of The Chain of Responsibility Pattern, 1st Par. – the intent of the Chain of Responsibility pattern is to provide a loosely coupled sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes the request along the chain until the request is handled; Sec. of The Chain of Responsibility Pattern, 2nd Par. – as is the case of our product support sample, the request may not be

fulfilled at one application site due to the distribution of the knowledge bases used to answer a request; the request must be routed until it can be handled).

16. **As to claim 23** (incorporating the rejection in claim 1) (Original), Thilmany discloses a (the) method which comprises providing files arranged to define the application specified therein, arranging the specification to be capable of deploying files and using the specification to deploy the files (e.g., Sec. of BizTalk® and the Orchestration Designer®, 3rd Par. – the Biztalk Orchestration Designer® is a Microsoft Visio® 2000-based design tool that, when compiled, can run as an XLANG® schedule; XLANG® itself is described through XML).

17. **As to claim 24** (incorporating the rejection in claim 23) (Original), Thilmany discloses a (the) method which comprises arranging the files specifying the application to be XML files (e.g., Sec. of BizTalk® and the Orchestration Designer®, 3rd Par. – the Biztalk Orchestration Designer® is a Microsoft Visio® 2000-based design tool that, when compiled, can run as an XLANG® schedule; XLANG® itself is described through XML).

18. **As to claim 25** (incorporating the rejection in claim 1) (Original), Thilmany discloses a (the) method in which the data processed by the application is specified in an XML file (e.g., Sec. of BizTalk® and the Orchestration Designer®, 3rd Par. – the Biztalk Orchestration Designer® is a Microsoft Visio® 2000-based design tool that,

when compiled, can run as an XLANG® schedule; XLANG® itself is described through XML).

19. **As to claim 26** (incorporating the rejection in claim 1) (Original), Thilmany discloses a (the) method in which data processed by the application is specified in an image file (e.g., Sec. of Biztalk® and the Orchestration Designer®, 3rd Par. – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of message quest, COM component, BizTalk® Channel, file, e-mail message, or HTTP-based service; with the Orchestration Designer, analysts not privy to the implementation complexities of COM or Web development can participate in the development of business rules by using the designer in Visio® to lay out the business flow; the developer can then implement the moving parts that these designers orchestrate).

20. **As to claim 28** (incorporating the rejection in claim 1) (Original), Thilmany discloses a (the) method which comprises providing a graphical tool arranged to enable a user to specify components of the application (e.g., Sec. of Biztalk® and the Orchestration Designer®, 3rd Par. – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of message quest, COM component, BizTalk® Channel, file, e-mail message, or HTTP-based service; with the Orchestration Designer, analysts not privy to the implementation complexities of COM or Web development can participate in the

development of business rules by using the designer in Visio® to lay out the business flow; the developer can then implement the moving parts that these designers orchestrate; Sec. of BizTalk® and the Orchestration Designer®, 3rd Par. – the Biztalk® Orchestration Designer® is a Microsoft Visio® 2000-based design tool that, when compiled, can run as an XLANG® schedule; XLANG® itself is described through XML).

21. **As to claim 29** (incorporating the rejection in claim 28) (Original), Thilmany discloses a (the) method which comprises providing a library of at least one of the following: nodes; node layers; specification; patterns; messages; rule sets; style sheets; schemas and in which the graphical tool allows a user to select components from one of said libraries (e.g., Sec. of Biztalk® and the Orchestration Designer®, 3rd Par. – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of message quest, COM component, BizTalk® Channel, file, e-mail message, or HTTP-based service; with the Orchestration Designer, analysts not privy to the implementation complexities of COM or Web development can participate in the development of business rules by using the designer in Visio® to lay out the business flow; the developer can then implement the moving parts that these designers orchestrate; SUMMARY – this article describes several traditional design patterns including the Observer pattern and the Dispatcher pattern, elaborates on their structures, what they're used for, and how they can help you build a Biztalk®-based solution; Sec. of The Chain of Responsibility Pattern, 1st Par. – the intent of the Chain of Responsibility pattern is to provide a loosely coupled

sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes the request along the chain until the request is handled; Sec. of The Chain of Responsibility Pattern, 2nd Par. – as is the case of our product support sample, the request may not be fulfilled at one application site due to the distribution of the knowledge bases used to answer a request; the request must be routed until it can be handled).

22. **As to claim 30** (incorporating the rejection in claim 29) (Original), Thilmany discloses a (the) method which allows a user to define further libraries (e.g., Sec. of Biztalk® and the Orchestration Designer®, 3rd Par. – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of message quest, COM component, BizTalk® Channel, file, e-mail message, or HTTP-based service; with the Orchestration Designer®, analysts not privy to the implementation complexities of COM or Web development can participate in the development of business rules by using the designer in Visio to lay out the business flow; the developer can then implement the moving parts that these designers orchestrate; Sec. of BizTalk® and the Orchestration Designer®, 3rd Par. – the Biztalk Orchestration Designer® is a Microsoft Visio® 2000-based design tool that, when compiled, can run as an XLANG® schedule; XLANG® itself is described through XML).

23. **As to claim 31** (incorporating the rejection in claim 28) (Original), Thilmany discloses a (the) method which comprises providing at least one pattern arranged to

define how nodes interact arranging the at least one pattern such that it is capable of interacting with at least one other pattern and arranging the graphical tool to allow a user to specify how the patterns and nodes interact with one another (e.g., SUMMARY – this article describes several traditional design patterns including the Observer pattern and the Dispatcher pattern, elaborates on their structures, what they're used for, and how they can help you build a Biztalk®-based solution; Sec. of The Chain of Responsibility Pattern, 1st Par. – the intent of the Chain of Responsibility pattern is to provide a loosely coupled sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes the request along the chain until the request is handled; Sec. of The Chain of Responsibility Pattern, 2nd Par. – as is the case of our product support sample, the request may not be fulfilled at one application site due to the distribution of the knowledge bases used to answer a request; the request must be routed until it can be handled).

24. **As to claim 32** (incorporating the rejection in claim 28) (Original), Thilmany discloses a (the) method which comprises using the graphical tool to perform at least one of the following: create the specification; edit the specification (e.g., Sec. of Biztalk® and the Orchestration Designer®, 3rd Par. – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of message quest, COM component, BizTalk® Channel, file, e-mail message, or HTTP-based service; with the Orchestration Designer, analysts not privy to the

implementation complexities of COM or Web development can participate in the development of business rules by using the designer in Visio to lay out the business flow; the developer can then implement the moving parts that these designers orchestrate; Sec. of BizTalk® and the Orchestration Designer®, 3rd Par. – the Biztalk Orchestration Designer® is a Microsoft Visio® 2000-based design tool that, when compiled, can run as an XLANG® schedule; XLANG® itself is described through XML).

25. **As to claim 33** (incorporating the rejection in claim 28) (Original), Thilmany discloses a (the) method which comprises using the graphical tool to manipulate any components of the specification (e.g., Sec. of Biztalk® and the Orchestration Designer®, 3rd Par. – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of message quest, COM component, BizTalk Channel, file, e-mail message, or HTTP-based service; with the Orchestration Designer®, analysts not privy to the implementation complexities of COM or Web development can participate in the development of business rules by using the designer in Visio® to lay out the business flow; the developer can then implement the moving parts that these designers orchestrate; Sec. of BizTalk® and the Orchestration Designer®, 3rd Par. – the Biztalk Orchestration Designer® is a Microsoft Visio® 2000-based design tool that, when compiled, can run as an XLANG® schedule; XLANG® itself is described through XML).

26. **As to claim 34** (incorporating the rejection in claim 1) (Original), Thilmany discloses a (the) method which comprises creating and deploying files and processing the files in the run time environment (e.g., Sec. of Biztalk® and the Orchestration Designer®, 3rd Par. – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of message quest, COM component, BizTalk® Channel, file, e-mail message, or HTTP-based service; with the Orchestration Designer®, analysts not privy to the implementation complexities of COM or Web development can participate in the development of business rules by using the designer in Visio to lay out the business flow; the developer can then implement the moving parts that these designers orchestrate; Sec. of BizTalk® and the Orchestration Designer®, 3rd Par. – the Biztalk Orchestration Designer® is a Microsoft Visio® 2000-based design tool that, when compiled, can run as an XLANG® schedule; XLANG® itself is described through XML).

27. **As to claim 35** (incorporating the rejection in claim 1) (Original), Thilmany discloses a (the) method in which at least one node is provided to provide an output from the application (e.g., Sec. of The Chain of Responsibility Pattern, 1st Par. – the intent of the Chain of Responsibility pattern is to provide a loosely coupled sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes the request along the chain until the request is handled; Sec. of The Chain of Responsibility Pattern, 2nd Par. – as is the case of our product support sample, the request may not be fulfilled at

one application site due to the distribution of the knowledge bases used to answer a request; the request must be routed until it can be handled).

28. **As to claim 36** (Currently amended), Thilmany discloses a computer system having a memory and being arranged to create an application, said system comprising:

- a node creation means creator arranged to create a definition specifying at least one node, and for each node specified, the definition defining how that node will interact with any other nodes, resources useable by that node (e.g., Sec. of the chain of responsibility pattern, 1st Par. – the intent of the Chain of Responsibility pattern is to provide a loosely coupled sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes the request along the chain until the request is handled; Sec. of Using the BizTalk Orchestration Designer®, 2nd Par. – ports are named locations where messages are sent and received; ports can be defined initially as unbound or bound), a set of predetermined rules to be used by that node for processing data (e.g., Sec. of Using the BizTalk Orchestration Designer®, 1st Par. – there are three key object categories in the business process page of the Orchestration Designer®; flowchart shapes, ports, and implementation shapes; conceptually, the way this works is that you, or your friendly neighborhood business analyst, use the flowchart shapes to create the process flow; the flowchart shapes support basic constructs such as if … then … else decisions, looping while a condition is true, branching execution, rejoining

the branches together, and defining transactional boundaries) and messages that can be passed between that node (e.g., Sec. of BizTalk® and the Orchestration Designer®, 2nd Par. – communicating to a BizTalk® server implementation simply means using XML as the message format; as you may already know, BizTalk® is completely structured around XML and uses a SOAP 1.1 XML message format; the BizTalk® friendly XML message is nothing more than a SOAP 1.1 XML message with additional biz-tags to comply with the BizTalk® Framework specification) and any other nodes such that that node is capable of receiving messages in any format which can trigger a rule if predetermined data is present, the node being capable of processing data according to a set of predetermined rules (e.g., Sec. of Using the BizTalk Orchestration Designer®, 1st Par. – there are three key object categories in the business process page of the Orchestration Designer®; flowchart shapes, ports, and implementation shapes; conceptually, the way this works is that you, or your friendly neighborhood business analyst, use the flowchart shapes to create the process flow; the flowchart shapes support basic constructs such as if … then … else decisions, looping while a condition is true, branching execution, rejoining the branches together, and defining transactional boundaries) and generating an output therefrom (e.g., Sec. of the chain of responsibility pattern, 1st Par. – the intent of the Chain of Responsibility pattern is to provide a loosely coupled sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes

the request along the chain until the request is handled; Sec. of Using the BizTalk Orchestration Designer, 2nd Par. – ports are named locations where messages are sent and received; ports can be defined initially as unbound or bound; Sec. of BizTalk® and the Orchestration Designer®, 3rd par. – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of a message queue, COM component, BizTalk Channel, e-mail message, or HTTP-based service; orchestration designer provides a way to create loosely coupled business processes that may optionally be long-running a nature);

- a linker capable of connecting at least two nodes such that data can pass between the nodes and being arranged to interact with the node creator to modify the definition; deployer arranged to deploy the application from the definition created by the node creator and the linker according to a specification (e.g., Sec. of the chain of responsibility pattern, 1st Par. – the intent of the Chain of Responsibility pattern is to provide a loosely coupled sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes the request along the chain until the request is handled; Sec. of Using the BizTalk Orchestration Designer®, 2nd Par. – ports are named locations where messages are sent and received; ports can be defined initially as unbound or bound);

29. **As to claim 37** (incorporating the rejection in claim 36) (Currently amended), Thilmany discloses a (the) computer system which comprises at least one processor arranged to process data, including the definition (e.g., Sec. of BizTalk® and the Orchestration Designer®, 3rd Par. – the Biztalk Orchestration Designer® is a Microsoft Visio® 2000-based design tool that, when compiled, can run as an XLANG® schedule; XLANG® itself is described through XML), and on which the definition created by the node creator and modified by the linker is processed (e.g., Sec. of the chain of responsibility pattern, 1st Par. – the intent of the Chain of Responsibility pattern is to provide a loosely coupled sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes the request along the chain until the request is handled; Sec. of Using the BizTalk Orchestration Designer®, 2nd Par. – ports are named locations where messages are sent and received; ports can be defined initially as unbound or bound).

30. **As to claim 38** (incorporating the rejection in claim 37) (Currently amended), Thilmany discloses a (the) computer system which comprises at least one processing apparatus comprising the at least one processor and in which the linker is arranged to connect nodes running on the processor within the processing apparatus (e.g., Sec. of the chain of responsibility pattern, 1st Par. – the intent of the Chain of Responsibility pattern is to provide a loosely coupled sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes the request along the chain until the request is handled; Sec.

of Using the BizTalk Orchestration Designer®, 2nd Par. – ports are named locations where messages are sent and received; ports can be defined initially as unbound or bound).

31. **As to claim 39** (incorporating the rejection in claim 38) (Currently amended), Thilmany discloses a (the) computer system which comprises a plurality of processors, each remote from the other and having connector therebetween capable of transmitting data between the processors (e.g., Sec. of BizTalk® and the Orchestration Designer®, 2nd Par. – communicating to a BizTalk® server implementation simply means using XML as the message format; as you may already know, BizTalk® is completely structured around XML and uses a SOAP 1.1 XML message format; the BizTalk® friendly XML message is nothing more than a SOAP 1.1 XML message with additional biz-tags to comply with the BizTalk® Framework specification).

32. **As to claim 40** (incorporating the rejection in claim 39) (Original), Thilmany discloses a (the) computer system in which each of the processors is provided on a separate processing apparatus (e.g., Sec. of The Chain of Responsibility Patter, 1st Par. – in cases where business rules may be distributed among separated applications or environments, it is prudent to hold requesters to a single relationship to avoid unnecessary coupling).

33. **As to claim 41** (incorporating the rejection in claim 39) (Currently amended), Thilmany discloses a (the) computer system in which the linker is arranged to connect nodes provided on processors remote from one another (e.g., Sec. of the chain of responsibility pattern, 1st Par. – the intent of the Chain of Responsibility pattern is to provide a loosely coupled sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes the request along the chain until the request is handled; Sec. of Using the BizTalk Orchestration Designer®, 2nd Par. – ports are named locations where messages are sent and received; ports can be defined initially as unbound or bound).

34. **As to claim 42** (incorporating the rejection in claim 36) (Currently amended), Thilmany discloses a (the) computer system in which the deployer deploys the definition that causes the nodes to communicate with one another using one of HTTP and direct memory protocols (e.g., Sec. of the chain of responsibility pattern, 1st Par. – the intent of the Chain of Responsibility pattern is to provide a loosely coupled sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes the request along the chain until the request is handled; Sec. of Using the BizTalk Orchestration Designer®, 2nd Par. – ports are named locations where messages are sent and received; ports can be defined initially as unbound or bound; Sec. of BizTalk and the Orchestration Designer®, 3rd par. – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of a message queue, COM

component, BizTalk Channel, e-mail message, or HTTP-based service; orchestration designer provides a way to create loosely coupled business processes that may optionally be long-running a nature).

35. **As to claim 43** (incorporating the rejection in claim 36) (Currently amended), Thilmany discloses a (the) computer system in which the node creation means creator is arranged to utilise at least one of the following: predetermined definitions and pre-written definitions (e.g., Sec. of Using the BizTalk Orchestration Designer®, 1st Par. – there are three key object categories in the business process page of the Orchestration Designer®; flowchart shapes, ports, and implementation shapes; conceptually, the way this works is that you, or your friendly neighborhood business analyst, use the flowchart shapes to create the process flow; the flowchart shapes support basic constructs such as if ... then ... else decisions, looping while a condition is true, branching execution, rejoining the branches together, and defining transactional boundaries).

36. **As to claim 44** (incorporating the rejection in claim 43) (Original), Thilmany discloses a computer system in which the pre-written definition is provided in at least one library (e.g., P. 1, 6th Par. – the sample application will show how applying patterns to Biztalk can significantly ease the development and design for each sub-department; it will provide the ability to answer and route online product support questions effectively and efficiently; Sec. of Design Patterns versus Architectural Patterns, 1st Par. – design patterns help make the design process faster; this allows solution providers to take the

time to concentrate on the business implementation; more importantly, patterns help formalize the design to make it reusable; reusability not only applies to the components themselves, but also the stages the design must go through to morph itself into your final solution; Sec. of Biztalk and the Orchestration Designer®, 3rd Par. – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of a message queue, COM component, BizTalk® Channel, file, e-mail message, or HTTP-based service; Orchestration Designer provides a way to create loosely coupled business processes that may optionally be long-running in nature).

37. **As to claim 45** (incorporating the rejection in claim 36) (Currently amended), Thilmany discloses a (the) computer system which further comprises a pattern creation means creator arranged to create at least one pattern of nodes (e.g., P. 1, 6th Par. – the sample application will show how applying patterns to Biztalk® can significantly ease the development and design for each sub-department; it will provide the ability to answer and route online product support questions effectively and efficiently; Sec. of Design Patterns versus Architectural Patterns, 1st Par. – design patterns help make the design process faster; this allows solution providers to take the time to concentrate on the business implementation; more importantly, patterns help formalize the design to make it reusable; reusability not only applies to the components themselves, but also the stages the design must go through to morph itself into your final solution; Sec. of Biztalk® and the Orchestration Designer, 3rd Par. – the designer provides a high-level

means of orchestrating a middleware system's moving parts; those moving parts can take the form of a message queue, COM component, BizTalk® Channel, file, e-mail message, or HTTP-based service; Orchestration Designer provides a way to create loosely coupled business processes that may optionally be long-running in nature).

38. **As to claim 46** (incorporating the rejection in claim 36) (Currently amended), Thilmany discloses a (the) computer system which further comprises a pattern cloner arranged to clone a pattern of nodes (e.g., P. 1, 6th Par. – the sample application will show how applying patterns to Biztalk can significantly ease the development and design for each sub-department; it will provide the ability to answer and route online product support questions effectively and efficiently; Sec. of Design Patterns versus Architectural Patterns, 1st Par. – design patterns help make the design process faster; this allows solution providers to take the time to concentrate on the business implementation; more importantly, patterns help formalize the design to make it reusable; reusability not only applies to the components themselves, but also the stages the design must go through to morph itself into your final solution; Sec. of Biztalk® and the Orchestration Designer®, 3rd Par. – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of a message queue, COM component, BizTalk® Channel, file, e-mail message, or HTTP-based service; Orchestration Designer provides a way to create loosely coupled business processes that may optionally be long-running in nature).

39. **As to claim 47** (incorporating the rejection in claim 36) (Currently amended), Thilmany discloses a (the) computer system which further comprises a rule creation means creator arranged to allow predetermined rules to be created and edited (e.g., Sec. of BizTalk® and the Orchestration Designer®, 3rd Par. – the Biztalk Orchestration Designer® is a Microsoft Visio® 2000-based design tool that, when compiled, can run as an XLANG® schedule; XLANG® itself is described through XML).

40. **As to claim 48** (incorporating the rejection in claim 36) (Currently amended), Thilmany discloses a (the) computer system which further comprises at least one of the following: a node storage (e.g., Sec. of BizTalk® and the Orchestration Designer®, 3rd Par. – the Biztalk Orchestration Designer® is a Microsoft Visio® 2000-based design tool that, when compiled, can run as an XLANG® schedule; XLANG® itself is described through XML); a pattern storage (e.g., P. 1, 6th Par. – the sample application will show how applying patterns to Biztalk can significantly ease the development and design for each sub-department; it will provide the ability to answer and route online product support questions effectively and efficiently); a rule storage (e.g., Sec. of Using the BizTalk Orchestration Designer®, 1st Par. – there are three key object categories in the business process page of the Orchestration Designer®; flowchart shapes, ports, and implementation shapes; conceptually, the way this works is that you, or your friendly neighborhood business analyst, use the flowchart shapes to create the process flow; the flowchart shapes support basic constructs such as if ... then ... else decisions,

looping while a condition is true, branching execution, rejoining the branches together, and defining transactional boundaries).

41. **As to claim 49** (incorporating the rejection in claim 1) (Original), refer to claim **1** above accordingly.

42. **As to claim 50** (incorporating the rejection in claim 36) (Original), refer to claim **36** above accordingly.

43. **As to claim 62** (incorporating the rejection in claim 1) (New), Thilmany discloses a (the) method in which the node, specification, and messages are at least in part written in XML (e.g., Sec. of BizTalk® and the Orchestration Designer®, 2nd Par. – communicating to a BizTalk® server implementation simply means using XML as the message format; as you may already know, BizTalk® is completely structured around XML and uses a SOAP 1.1 XML message format; the BizTalk® friendly XML message is nothing more than a SOAP 1.1 XML message with additional biz-tags to comply with the BizTalk® Framework specification).

44. **As to claim 63** (New), Thilmany discloses a computer system having a memory and being arrange to run an application, said system comprising:

- a run time environment arranged to process at least one machine readable data file, the machine readable data file providing a definition of at least one node and

a specification, the specification defining how the at least one node can interact with other nodes (e.g., Sec. of the chain of responsibility pattern, 1st Par. – the intent of the Chain of Responsibility pattern is to provide a loosely coupled sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes the request along the chain until the request is handled; Sec. of Using the BizTalk Orchestration Designer®, 2nd Par. – ports are named locations where messages are sent and received; ports can be defined initially as unbound or bound), resources useable by the at least one node, at least one set of predetermined rules to be used by the node for processing data and messages (e.g., Sec. of Using the BizTalk Orchestration Designer®, 1st Par. – there are three key object categories in the business process page of the Orchestration Designer®; flowchart shapes, ports, and implementation shapes; conceptually, the way this works is that you, or your friendly neighborhood business analyst, use the flowchart shapes to create the process flow; the flowchart shapes support basic constructs such as if … then … else decisions, looping while a condition is true, branching execution, rejoining the branches together, and defining transactional boundaries) that can be passed between nodes such that a node is capable of receiving messages in any format which can trigger a rule if predetermined data is present (e.g., Sec. of BizTalk® and the Orchestration Designer®, 2nd Par. – communicating to a BizTalk® server implementation simply means using XML as the message format; as you may

already know, BizTalk® is completely structured around XML and uses a SOAP 1.1 XML message format; the BizTalk® friendly XML message is nothing more than a SOAP 1.1 XML message with additional biz-tags to comply with the BizTalk® Framework specification);

- wherein the run time environment comprises a linker which is arranged to connect the at least one node to any other nodes such that data input to the application is processed by the at least one node and, if further processing is required, forwarded to the other nodes for that further processing (e.g., Sec. of the chain of responsibility pattern, 1st Par. – the intent of the Chain of Responsibility pattern is to provide a loosely coupled sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes the request along the chain until the request is handled; Sec. of Using the BizTalk Orchestration Designer®, 2nd Par. – ports are named locations where messages are sent and received; ports can be defined initially as unbound or bound).

45. **As to claim 64** (incorporating the rejection in claim 63) (New), refer to claim **63** above accordingly.

***Claim Rejections – 35 USC § 103(a)***

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

46. Claims 10-12, and 14 are rejected under 35 U.S.C. 103(a) as being unpatentable over Thilmany in view of Moore et al., (Pub. No. US 2004/0034848 A1 B1) (hereinafter 'Moore' - art made of record)

47. **As to claim 10** (incorporating the rejection in claim 4) (Original), Thilmany does not explicitly disclose a (the) method which comprises arranging at least one of the layers of a node to act as a rule processing engine arranged to apply the predetermined rules to data that the node receives.

However, in an analogous art of Rule Engine, Moore discloses a (the) method which comprises arranging at least one of the layers of a node to act as a rule processing engine arranged to apply the predetermined rules to data that the node receives (e.g., ., Fig. 1 – element of 140 – Business Intelligence Server; [0031], Lines 16-18 – rule-packs are implemented as extensible markup language (XML) documents expressed in a rules markup language generated for that purpose).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Moore into the Thilmany's system to further provide a (the) method which comprises arranging at least one of the layers of a node to act as a rule processing engine arranged to apply the predetermined rules to data that the node receives in Thilmany system.

The motivation is that it would further enhance the Thilmany's system by taking, advancing and/or incorporating Moore's system which offers significant advantages for automating business processes including, in a computer system, receiving a rule set as a single package, determining logical conflicts within the rule set, resolving the logical conflicts, and generating a sequence of processing logic from the rule set for optimal processing of inputted facts as once suggested by Moore (e.g., [0005]).

48. **As to claim 11** (incorporating the rejection in claim 10) (Currently amended), Moore discloses a (the) method in which the rule processing engine layer of a node is arranged to use uses forward chaining rule logic (e.g., [0002], Lines 8-12 – forward chaining is a process of applying a set of previously determined rules to the facts in a knowledge base to see if any of them fire and thereby generate new facts; in essence, all derivable knowledge is derived in forward chaining).

49. **As to claim 12** (incorporating the rejection in claim 10) (Original), Moore discloses a method which comprises providing a rule set of at least one rule in a file that is used by the rule processing engine (e.g., [0005], automating business processes including, in a computer system, receiving a rule set as a single package, determining logical conflicts within the rule set, resolving the logical conflicts, and generating a sequence of processing logic from the rule set for optimal processing of inputted facts).

50. **As to claim 14** (incorporating the rejection in claim 10) (Original), Moore discloses a (the) method which comprises defining each rule set that is to be used by the application in the specification (e.g., [0031], Lines 16-18 – rules-packs are implemented as extensible markup language (XML) documents expressed in a rules markup language generated for that purpose).

51. Claim 13 is rejected under 35 U.S.C. 103(a) as being unpatentable over Thilmany in view of Moore and further in view of Lee et al. (*The Extensible Rule Markup Language, May 2003, ACM*) (hereinafter 'Lee' - art made of record)

52. **As to claim 13** (incorporating the rejection in claim 12) (Original), Thilmany and Moore do not explicitly disclose a (the) method which comprises specifying the file in which the rules are located by a link.

However, in an analogous art of The Extensible Rule Markup Language, Lee discloses a (the) method which comprises specifying the file in which the rules are located by a link (e.g., P. 60, Left-Col, "Relevance Linkability" – Linkages of the relevance between hypertexts with RIML, as well as rules in RSML syntax (called RSML rules), should be expressed completely) .

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Lee into the Thilmany-Moore's system to further provide a (the) method which comprises specifying the file in which the rules are located by a link.

The motivation is that it would further enhance the Thilmany-Moore's system by taking, advancing and/or incorporating Lee's system which offers significant advantages for the Extensible Markup Language (XML) explicates the implicitly embedded data in a formal structure with mutually agreed-on semantic definitions; may industrial-strength standard initiatives using XML are under way, including the Electronic Business XML Initiative, XML Common Business Library, Open Trading Protocol, Open Business on the Internet, Common Business Language, RosettaNet, and Biztalk as once suggested by Capra (e.g., P. 59, 1st Par.).

53. Claims 17 and 17 are rejected under 35 U.S.C. 103(a) as being unpatentable over Thilmany in view of Bowman-Amuah (hereinafter 'Bowman-Amuah') (Pat. No. US 6,601,234 B1).

54. **As to claim 17** (incorporating the rejection in claim 1) (Currently amended), Thilmany discloses a (the) method which comprises writing the messages in a flat text format, which may be any of the following: XML (e.g., Sec. of Biztalk® and the Orchestration Designer®, 2nd Par. – communicating to a BizTalk® server implementation simply means using XML as the message format).

But, Thilmany does not explicitly disclose a (the) method which comprises writing the messages in a flat text format, which may be any of the following: ASCII, EDI (Electronic Data Interchange).

However, in an analogous art of Attribute Dictionary in a Business Logic Services Environment, Bowman-Amuah discloses a (the) method which comprises writing the messages in a flat text format, which may be any of the following: ASCII (e.g., Col. 56, Lines 55-60), EDI (Electronic Data Interchange) (Col. 78, Line 35 through Col. 79, Line 9).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Bowman-Amuah into the Thilmany's system to further provide a (the) method which comprises writing the messages in a flat text format, which may be any of the following: ASCII, EDI (Electronic Data Interchange).

The motivation is that it would further enhance the Thilmany's system by taking, advancing and/or Bowman-Amuah system which offers significant advantages for a system and method providing for controlling access to data of a business object via an attribute dictionary; the attribute dictionary, which stores attribute names and values, is dispatches over a network as once suggested by Bowman-Amuah (e.g., Abstract).

55. **As to claim 27** (incorporating the rejection in claim 1) (Original), refer to claim **17** above accordingly.

***Conclusion***

56. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-

1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

TUAN DAM
SUPERVISORY PATENT EXAMINER

BCW

September 10, 2007